Network Working Group A. Shpiner
Internet Draft Technion - Israel Institute of Technology
Intended status: Experimental R. Tse
Expires: April 2013 C. Schelp
PMC-Sierra
T. Mizrahi
Marvell
October 15, 2012

# Multi-Path Time Synchronization
**draft-shpiner-multi-path-synchronization-00.txt**

Abstract

Clock synchronization protocols are very widely used in IP-based networks. The Network Time Protocol (NTP) has been commonly deployed for many years, and the last few years have seen an increasingly rapid deployment of the Precision Time Protocol (PTP). As time-sensitive applications evolve, clock accuracy requirements are becoming increasingly stringent, requiring the time synchronization protocols to provide high accuracy. Slave Diversity is a recently introduced approach, where the master and slave clocks (also known as server and client) are connected through multiple network paths, and the slave combines the information received through all paths to obtain a higher clock accuracy compared to the conventional one-path approach. This document describes a multi-path approach to PTP and NTP over IP netwokrs, allowing the protocols to run concurrently over multiple communication paths between the master and slave clocks. The multi-path approach can significantly contribute to clock accuracy, security and fault protection. The Multi-Path Precision Time Protocol (MPPTP) and Multi-Path Network Time Protocol (MPNTP) define an additional layer that extends the existing PTP and NTP without the need to modify these protocols. MPPTP and MPNTP also allow backward compatibility with nodes that do not support the multi-path extension.

Table of Contents

## 1. Introduction

The two most common time synchronization protocols in IP networks are the Network Time Protocol [NTP], and the Precision Time Protocol (PTP), defined in the IEEE 1588 standard [IEEE1588].
The accuracy of the time synchronization protocols directly depends on the stability and the symmetry of propagation delays on both directions between the master and slave clocks. Depending on the nature of the underlying network, time synchronization protocol packets can be subject to variable network latency or path asymmetry (e.g. [ASSYMETRY], [ASSYMETRY2]). As time sensitive applications evolve, accuracy requirements are becoming increasingly stringent.

Using a single network path in a clock synchronization protocol closely ties the slave clock accuracy to the behavior of the specific path, which may suffer from temporal congestion, faults or malicious attacks. Relying on multiple clock servers as in NTP solves these problems, but requires active maintenance of multiple accurate sources in the network, which is not always possible. The usage of Transparent Clocks (TC) in PTP solves the congestion problem by eliminating the queueing time from the delay calculations, but requires the intermediate routers and switches to support the TC functionality, which is not always the case.

```
                                  ____
                           ______/    \_____
                       ___/                 \____
                  ____/                          \
      ____       /           path 1              /           ___
     /    \     /    ________________________    \          /   \
    /Master\____\___/                        \____\________/Slave\
    \Clock /    /   \________ _______________/     \       \Clock/
     \____/     \                                  /        \__ /
                 \____       path 2             __/
                      \_______           ______/
                              \_________/
```

Figure 1 Multi-Path Connection

Since master and slave clocks are often connected through more than one path in the network, as shown in Figure 1, [SLAVEDIV] suggested that a time synchronization protocol can be run over multiple paths, providing several advantages. First, it can significantly increase the clock accuracy as shown in [SLAVEDIV]. Second, this approach provides additional security, allowing to mitigate man-in-the-middle attacks against the time synchronization protocol [DELAY-ATT]. Third, using multiple paths concurrently provides an inherent failure protection mechanism with a negligible recovery time.

This document introduces Multi-Path PTP (MPPTP) and Multi-Path NTP (MPNTP), respectively. These extensions are defined at the network layer, and do not require any changes in the PTP or in the NTP protocols.

MPPTP and MPNTP are defined over IP networks. As IP networks typically combine ECMP routing, this property is leveraged for the multiple paths used in MPPTP and MPNTP. The key property of the multi-path extensions is that clocks in the network can use more than one IP address. Each {master IP, slave IP} address pair defines a path. Depending on the network topology and configuration, the IP combination pairs can form multiple diverse paths used by the multi-path synchronization protocols.

This document introduces two variants for each of the two multi-path protocols; a variant that requires all nodes to support the multi-path protocol, referred to as the two-way variant, and a backward compatible variant that allows a multi-path clock to connect to a conventional single-path clock, referred to as the one-way variant.

## 2. Conventions Used in this Document

### 2.1. Abbreviations

ECMP   Equal Cost Multiple Path

LAN    Local Area Network

MPNTP   Multi-Path Network Time Protocol

MPPTP   Multi-Path Precision Time Protocol

NTP    Network Time Protocol

PTP    Precision Time Protocol

### 2.2. Terminology

In the NTP terminology, a time synchronization protocol is run between a client and a server, while PTP uses the terms master and slave. Throughout this document, the sections that refer to both PTP and NTP generically use the terms master and slave.

## 3. Multiple Paths in IP Networks

### 3.1. Load Balancing

Traffic sent across IP networks is often load balanced across multiple paths. The load balancing decisions are typically based on packet header fields: source and destination addresses, Layer 4 ports, the Flow Label field in IPv6, etc.
Three common load balancing criteria are per-destination, per-flow and per-packet. The per-destination load balancers take a load balancing decision based on the destination IP address. Per-flow load balancers use various fields in the packet header, e.g., IP addresses and Layer 4 ports, for the load balancing decision. Per-packet load balancers use flow-blind techniques such as round-robin without basing the choice on the packet content.

### 3.2. Using Multiple Paths Concurrently

To utilize the diverse paths that traverse per-destination load-balancers or per-flow load-balancers, the packet transmitter can vary the IP addresses in the packet header. The analysis in [PARIS2] shows that a significant majority of the flows on the internet traverse per-destination or per-flow load-balancing. It presents statistics that 72% of the flows traverse per-destination load balancing and 39% of the flows traverse per-flow load-balancing, while only a negligible part of the flows traverse per-packet load balancing. These statistics show that the vast majority of the traffic on the internet is load balanced based on packet header fields.

The approaches in this draft are based on varying the source and destination IP addresses in the packet header. Possible extensions have been considered that also vary the UDP ports. However some of the existing implementations of PTP and NTP use fixed UDP port values in both the source and destination UDP port fields, and thus do not allow this approach.

### 3.3. Two-Way Paths

A key property of IP networks is that packets forwarded from A to B do not necessarily traverse the same path as packets from B to A. Thus, we define a two-way path for a master-slave connection as a pair of one-way paths: the first from master to slave and the second from slave to master.

In a locally administered network, a traffic engineering approach can be used to verify that time synchronization traffic is always forwarded through bidirectional two-way paths, i.e., that each two way path uses the same route on the forward and reverse directions. However, in the general case two-way paths do not necessarily use the same path for the forward and reverse directions.

## 4. Solution Overview

The multi-path time synchronization protocols we present are comprised of two building blocks; one is the path configuration and identification, and the other is the algorithm used by the slave to combine the information received from the various paths.

### 4.1. Path Configuration and Identification

The master and slave clocks must be able to determine the path of transmitted protocol packets, and to identify the path of incoming protocol packets. A path is determined by a {master IP, slave IP} address pair. The synchronization protocol message exchange is run independently through each path.

Each IP address pair defines a two-way path, and thus allows the clocks to bind a transmitted packet to a specific path, or to identify the path of an incoming packet.

In locally administered IP networks, the routing tables across the network can be configured with multiple traffic engineered paths between the pair of clocks. By carefully configuring the routers in such networks it is possible to create diverse paths for each of the IP address pairs between two clocks in the network. However, in public and provider networks the load balancing behavior is hidden from the end users. In this case the actual number of paths may be less than the number of IP address pairs, since some of the address pairs may share common paths.

### 4.2. Combining

Various methods can be used for combining the time information received from the different paths. This document surveys several combining methods in Section 6 . The output of the combining algorithm is the accurate time offset.

## 5. Multi-Path Time Synchronization Protocols over IP Networks

This section presents two variants of MPPTP and MPNTP; one-way multi-path time synchronization and two-way multi-path time synchronization. In the first variant the multi-path protocol is run only by the slave, and the master is not aware of its usage. In the second variant all clocks must support the multi-path protocol.

The two-way protocol provides higher path diversity by using multiple IP addresses at both ends, the master and slave, while the one-way protocol only uses multiple addresses at the slave. On the other hand, the two-way protocol can only be deployed in networks where all the clocks support this protocol, while the one-way protocol can be used in hybrid networks.

Multi-path time synchronization, in both variants, requires clocks to use multiple IP addresses. This approach introduces a tradeoff; using a large number of IP addresses allows a large number of diverse paths, providing the advantages of slave diversity discussed in Section 1. On the other hand, a large number of IP addresses is more costly, requires the network topology to be more redundant and yields a management overhead.

The descriptions in this section refer to the end-to-end scheme of PTP, but are similarly applicable to the peer-to-peer scheme. The MPNTP protocol described in this document refers to the NTP client-server mode, although the concepts described here can be extended to include the symmetric variant as well.

Multi-path synchronization protocols by nature require protocol messages to be sent as unicast. Specifically in PTP, the following messages must be sent as unicast in MPPTP: Sync, Delay_Req, Delay_Resp, PDelay_Req, PDelay_Resp, Follow_Up, and PDelay_Resp_Follow_Up. Note that [IEEE1588] allows these messages to be sent either as multicast or as unicast.

### 5.1. One-Way Multi-Path Synchronization

In the one-way approach, only the slave is aware of the fact that multiple paths are used, while the master is agnostic to the usage of multiple paths. This approach allows a hybrid network, where some of the clocks are multi-path clocks, and others are conventional one-path clocks. A one-way multi-path clock presents itself to the network as N independent clocks, using N IP addresses, and N clock identity values. Thus, the usage of multiple slave identities by a slave clock is transparent from the master's point of view, such that it treats each of the identities as a separate slave clock.

#### 5.1.1. One-Way MPPTP Synchronization Message Exchange

The one-way MPPTP message exchange procedure is as follows.

- o Each one-way MPPTP clock has a fixed set of N IP addresses and N corresponding clockIdentities . One of the IP addresses and clockIdentity values are defined as the clock primary identity.

- o The BMC algorithm determines the master.

- o Every one-way MPPTP port that is not in the 'slave' state (i.e., either a master or a clock that has just joined the network) periodically sends Announce messages using its primary identity.

- o A one-way MPPTP clock that is in the 'slave' state periodically transmits a set of N Announce messages using its N identities, while a clock in the 'master' state only transmits Announce messages using its primary identity.

- o The master periodically sends unicast Sync messages from its primary identity address to each of the slaves identified by the sourcePortIdentity and IP address.

- o The slave, upon receiving a Sync message, identifies its path according to the destination IP address. In response to the Sync message the slave sends a Delay_Req unicast message to the primary identity of the master. This message is sent using the slave identity corresponding to the path the Sync was received through.

- The master, in response to a Delay_Req message from the slave, responds with a Delay_Resp message using the IP address and sourcePortIdentity from the Delay_Req message.

- Upon receiving the Delay_Resp message, the slave identifies the path using the destination IP address and the requestingPortIdentity. The slave can then compute the corresponding path delay and the offset from the master.

### 5.1.2. One-Way MPNTP Synchronization Message Exchange

The one-way MPNTP message exchange procedure is as follows.

- A one-way MPNTP client has N separate identities, i.e., N IP addresses, and N corresponding Reference IDs.

- A one-way MPNTP client initiates the NTP protocol with an NTP server N times, using each of its N identities.

- The NTP protocol is maintained between the server and each of the N client identities.

- The client sends NTP messages to the master using each of its N identities.

- The server responds to the client's NTP messages using the IP address from the received NTP packet.

- The client, upon receiving an NTP packet, uses the IP destination address to identify the path it came through, and uses the time information accordingly.

### 5.2. Two-Way Multi-Path Synchronization

In two-way multi-path synchronization each clock has N IP addresses. Time synchronization messages are exchanged between each combination of {master IP, slave IP} addresses, allowing multiple paths between the master and slave.  Note that the actual number of paths between the master and slave may be less than the number of {master, slave} IP address pairs.

Once the multiple two-way connections are established, a separate synchronization protocol exchange instance is run through each of them.

### 5.2.1. Two-Way MPPTP Synchronization Message Exchange

The two-way MPPTP message exchange procedure is as follows.

- o Every clock periodically sends a set of N Announce messages, using its N addresses as the source IP address. The sourcePortIdentity field in the PTP header remains the same for all PTP messages of a given clock.

- o The BMC algorithm determines the master. Clocks are identified by the sourcePortIdentity and not by the IP address.

- o Each clock learns the multiple IP addresses of other clocks from the source IP addresses of the Announce packets it receives.

- o The master periodically sends unicast Sync messages from each of its N_m IP addresses to each of the slave's N_s IP addresses.

- o The slave, upon receiving a Sync message, identifies its path according to the {source, destination} IP addresses. In response to the Sync message the slave sends a Delay_Req unicast message, swapping the source and destination IP addresses from the Sync message.

- o The master, in response to a Delay_Req message from the slave, responds with a Delay_Resp message using the sourcePortIdentity from the Delay_Req message, and swapping the IP addresses from the Delay_Req.

- o Upon receiving the Delay_Resp message, the slave identifies the path using the {source, destination} IP address pair. The slave can then compute the corresponding path delay and the offset from the master.

- o The PTP protocol messages are sent through each of the N_m*N_s paths, and the slave combines the information from all these paths.

### 5.2.2. Two-Way MPNTP Synchronization Message Exchange

The MPNTP message exchange procedure is as follows.

- o Each NTP clock has a set of N IP addresses. The assumption is that the server information, including its multiple IP addresses is known to the NTP clients.

- o The MPNTP client initiates the N_s*N_c instances of the protocol, one for each {server IP, client IP} pair, allowing the client to combine the information from the N_s*N_c paths.
(N_s and N_c indicate the number of IP addresses used by the server and client, respectively.)

- o The client sends NTP messages to the master using each of the source-destination address combinations.

- o The server responds to the client's NTP messages using the IP address combination from the received NTP packet.

- o Using the {source, destination} IP address pair in the received packets, the client identifies the path, and performs its computations for each of the paths accordingly.

### 5.3. Using Traceroute for Path Discovery

The protocols presented above use multiple IP addresses in a single clock to create multiple paths. However, although each two-way path is defined by a different {master, slave} address pair, some of the IP address pairs may traverse exactly the same network path, making them redundant. Traceroute-based path discovery can be used for filtering only the IP addresses that obtain diverse paths. 'Paris Traceroute' [PARIS] and 'TraceFlow' [TRACEFLOW] are examples of tools that discover the paths between two points in the network.

The Traceroute-based filtering can be implemented by both master and slave nodes, or it can be restricted to run only on slave nodes to reduce the overhead on the master.

## 6. Combining Algorithm

Previous sections discussed the methods of creating the multiple paths and obtaining the time information required by the slave algorithm. This section discusses the algorithm used to combine this information into a single accurate time estimate. Note that the choice of the combining algorithm is local to the slave, and does not affect the interoperability of the protocol.
Several combining methods are examined next.

### 6.1. Averaging

In the first method the slave performs an autonomous time computation for each of the master-slave paths, and obtains the combined time by simply averaging the separate instances. This method can be further enhanced by adding weights to each of the paths. For example, a reasonable weighting choice is to use an inverse of the round-trip delay between the peers. Another option is to use the inverse of the path delay variance. , which is approximately the maximum likelihood estimator under certain assumptions [WEIGHT-MEAN].

### 6.2. Switching / Dynamic Algorithm

The switching and dynamic algorithms are presented in [SLAVEDIV]. The switching algorithm periodically chooses a primary path, and performs all time computations based on the protocol packets received through the primary path. The primary path is defined as the path with the minimal distance between the sampled delay and the average delay. The dynamic algorithm dynamically chooses between the result of the switching algorithm and the averaging.

### 6.3. NTP-like Filtering-Clustering-Combining Algorithm

NTP ([NTP], [NTP2]) provides an efficient algorithm of combining offset samples from multiple peers. The same approach can be used in MPPTP and MPNTP.

In the MPNTP, the selection and combining algorithms treat the offset samples from multiple paths as NTP treats samples from distinct peers. The rest of the selection and combining algorithms, as well as clock control logic is the same as in conventional NTP. In MPPTP, a similar approach to NTP can be adopted.

The combining algorithm [NTP3] contains three steps: filtering, selection and clustering.

In the filtering step, the best of the last n (usually n=8) samples of each peer is chosen. The choice criterion is the combination of a round trip delay estimate of the sample and the distance from the average offset of all n samples of a peer.

In the selection step the peers are divided into two groups: true-chimers and false tickers.

The clustering step chooses a subset of the true-chimers, whose peer jitter (the variance of peer offset samples) is smaller than the total select jitter of all selected peer offsets (the variance of the best offset of the selected peers).

The offset samples that passed through the three steps are combined by a weighted average into a single offset estimate. Detailed explanations are provided in [NTP2],[NTP3].

## 7. Security Considerations

The security aspects of time synchronization protocols are discussed in detail in [TICTOCSEC]. The methods describe in this document propose to run a time synchronization protocol through redundant paths, and thus allow to detect and mitigate man-in-the-middle attacks, as described in [DELAY-ATT].

## 8. IANA Considerations

There are no IANA actions required by this document.

RFC Editor: please delete this section before publication.

## 9. Acknowledgments

This document was prepared using 2-Word-v2.0.template.dot.

## 10. References

### 10.1. Normative References

[IEEE1588]    IEEE Instrumentation and Measurement Society, "IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems", IEEE Std 1588, 2008.

[NTP]    D. Mills, J. Martin, J. Burbank, W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", IETF, RFC 5905, 2010.

### 10.2. Informative References

[ASSYMETRY]    Yihua He and Michalis Faloutsos and Srikanth Krishnamurthy and Bradley Huffaker, "On routing asymmetry in the internet", IEEE Globecom, 2005.

[ASSYMETRY2]    Abhinav Pathak, Himabindu Pucha, Ying Zhang, Y. Charlie Hu, and Z. Morley Mao, "A measurement study of internet delay asymmetry", PAM'08, 2008.

[DELAY-ATT]    T. Mizrahi, "A Game Theoretic Analysis of Delay Attacks against Time Synchronization Protocols", ISPCS, 2012.

[NTP2]        Mills, D.L., "Internet time synchronization: the Network Time Protocol", IEEE Trans. Communications COM-39, 10 (October 1991), 1482-1493.

[NTP3]        Mills, D.L., "Improved algorithms for synchronizing computer network clocks", IEEE/ACM Trans. Networks 3, 3(June 1995), 245-254.

[PARIS]       Brice Augustin, Timur Friedman and Renata Teixeira, "Measuring Load-balanced Paths in the Internet", IMC, 2007.

[PARIS2]      B. Augustin, T. Friedman, and R. Teixeira, "Measuring Multipath Routing in the Internet", IEEE/ACM Transactions on Networking, 19(3), p. 830 - 840, June 2011.

[SLAVEDIV]   T. Mizrahi, "Slave Diversity: Using Multiple Paths to Improve the Accuracy of Clock Synchronization Protocols", ISPCS, 2012.

[TICTOCSEC]   T. Mizrahi, K. O'Donoghue, "TICTOC Security Requirements", IETF, draft-ietf-tictoc-security-requirements, work in progress, 2012.

[TRACEFLOW]   J. Narasimhan, B. V. Venkataswami, R. Groves and P. Hoose, "Traceflow", IETF, draft-janapath-intarea-traceflow, work in progress, 2012.

[WEIGHT-MEAN] http://en.wikipedia.org/wiki/Weighted_mean#Dealing_wi th_variance

## Authors' Addresses

Alex Shpiner
Department of Electrical Engineering
Technion - Israel Institute of Technology
Haifa, 32000 Israel

Email: shalex@tx.technion.ac.il

Richard Tse
PMC-Sierra
8555 Baxter Place
Burnaby, BC
Canada
V5A 4V7

Email: Richard.Tse@pmcs.com

Craig Schelp
PMC-Sierra
8555 Baxter Place
Burnaby, BC
Canada
V5A 4V7

Email: craig.schelp@pmcs.com

Tal Mizrahi
Marvell
6 Hamada St.
Yokneam, 20692 Israel

Email: talmi@marvell.com